

# Zerotree Design for Image Compression: Toward Weighted Universal Zerotree Coding

Michelle Effros \*

Department of Electrical Engineering (136-93)  
California Institute of Technology  
Pasadena, California 91125  
effros@caltech.edu

## Abstract

*We consider the problem of optimal, data-dependent zerotree design for use in weighted universal zerotree codes for image compression. A weighted universal zerotree code (WUZYC) is a data compression system that replaces the single, data-independent zerotree of Said and Pearlman with an optimal collection of zerotrees for good image coding performance across a wide variety of possible sources. We briefly describe the weighted universal zerotree encoding and design algorithms but focus primarily on the problem of optimal, data-dependent zerotree design. We demonstrate the performance of the proposed algorithm by comparing, at a variety of target rates, the performance of a Said-Pearlman style code using the standard zerotree to the performance of the same code using a zerotree designed with our algorithm. The comparison is made without entropy coding. The proposed zerotree design algorithm achieves, on a collection of combined text and gray-scale images, up to 4 dB performance improvement over a Said-Pearlman zerotree.*

## 1 Introduction

Data compression systems are redundancy removing systems. As a result, assumptions or a priori knowledge about the types of data to be compressed play a crucial role in the design of most compression algorithms. Sometimes, this role is explicit – as in the design of a vector quantizer, where the designer gathers a set of training data and uses that data in the design of an appropriate collection of code-words. In other algorithms, the assumptions employed by the code's designer may be less obvious, but nonetheless implicit in the code structure. JPEG's use of the DCT, for example, suggests an implicit assumption that the images to be coded will be from the class of "smooth, natural" images for which the DCT serves as an effective decorrelating transform.

---

\*This material is based upon work supported by NSF Grant No. MIP-9501977.

Like these other algorithms, the zerotree algorithms of Shapiro [1] and Said and Pearlman [2] rely upon a variety of assumptions about the types of data to be compressed. In particular, the designers assume first that most of the energy in the images to be compressed will be found in the lower frequency bands of those images' wavelet decompositions (another smooth image assumption). Second, the designers assume that the high energy values that do appear in higher frequency bands will usually be predictable by high energy values in the corresponding coefficients from lower frequency bands. In terms of compression performance, the first of these properties plays the greater role. That is, while the performance of zerotree codes on data sets that meet the first but not the second condition is still quite good, zerotree codes perform poorly on data sets that meet only the second of our two conditions.

When the assumptions upon which we build a compression system fail, performance suffers. Just as a vector quantizer designed for one source and operated on another will not achieve the optimal performance on the source in operation, a zerotree code used to compress an image that has most of its energy in the high frequency bands, for example, will yield poor performance, and can even lead to data expansion rather than compression.

We are interested in compression systems that are used to compress a variety of different images. To guarantee good performance across a wide class of possible data types, we can replace any single code with a family of codes. Given a collection of possible source codes, the optimal performance achievable with that collection is the performance associated with compressing each data block with the code from our collection that best matches the statistics of that block. We will call a code that achieves this performance an *omniscient* code, since both the encoder and decoder of such a code must somehow divine the optimal code for each data block and use that knowledge to switch appropriately among the codes in the collection.

Unfortunately, practical source codes cannot be omni-

scient. In a real compression system, the encoder has access to the uncoded data, and thus can effectively choose the best strategy for the data to be coded. The decoder, however, has no basis upon which to choose the optimal code, and thus a real decoder cannot be omniscient. As a result, real data compression systems that optimally switch strategies to match the statistics of an unknown or time-varying data source must use overhead bits to describe to the decoder which code will be employed on each data block. We call such a switched coding strategy a *two-stage source code* since the data is described in two sections or stages. In the first-stage description, the encoder describes a code from its collection. In the second-stage description, the encoder describes the data using the code described in the first stage.

A universal source code is a single data compression algorithm that asymptotically achieves the optimal performance on every source in some broad class of possible sources. Most universal source codes in the literature can be described in the above two-stage coding paradigm. Universal source coding theory has contributed both to the understanding of optimal coding performance and to the design of practical codes for image compression. Using this theory, we gain insight into the design of collections of a wide variety of types of codes including vector quantizers, quantization matrices for JPEG-style codes, and transform codes [3, 4, 5, 6, 7].

We here consider the use of universal source coding techniques to design a universal zerotree code, a collection of zerotree codes that achieves, on every source in some broad class of possible sources, performance approaching the performance that would be achieved if the optimal zerotree were employed on every data block. While [1] and [2] provide examples of good zerotrees for the set of natural images, they do not describe a general method for designing zerotrees for other classes of images. Thus in order to develop a weighted universal zerotree code (WUZYC), we must understand the design and operation of both two-stage codes and single zerotree codes. Section 2 contains a description of the optimal WUZYC encoding and design algorithms. Section 3 treats the design of a single zerotree. Section 4 contains experimental results.

## 2 Weighted Universal Zerotree Coding

In traditional zerotree coding, we use a wavelet decomposition and quadtree to impose an ordering on a collection of image coefficients. At the root of this quadtree are placed coefficients from the lowest frequency image subband, below them are the next higher bands, below them even higher bands, and so on. Image coefficients within this structure are described one band at a time, with the encoding process traveling only as deep in the given tree

structure as necessary to describe all values above an octavely decreasing threshold. The zerotree description is most efficient when the smallest subtree containing all coefficients below a given threshold is as shallow as possible, or equivalently, when that smallest subtree contains as few below-threshold nodes as possible.

We use  $x^l = (x_1, \dots, x_l) \in \mathcal{X}^l$  to represent an  $l$ -dimensional data vector of wavelet packet coefficients. Given an encoding algorithm like the one described in [2], associated with any zerotree description  $Z$  is a quantizer  $C = \beta \circ \alpha$  with encoder  $\alpha: \mathcal{X}^l \rightarrow \mathcal{S}$  and decoder  $\beta: \mathcal{S} \rightarrow \hat{\mathcal{X}}^l$  that together map the input space  $\mathcal{X}^l$  of possible data vectors to the output space  $\hat{\mathcal{X}}^l$  of possible reproductions by way of a binary prefix code  $\mathcal{S}$ . (We will associate with each zerotree a target final rate, and thus assume that  $\alpha(x^l)$ , or more explicitly  $|\alpha(x^l)|$  is a constant for any  $x^l \in \mathcal{X}^l$ .) Let

$$d(x^l, Z) = d(x^l, \beta(\alpha(x^l)))$$

and

$$r(x^l, Z) = |\alpha(x^l)|$$

be the distortion and rate respectively achieved by quantizing  $x^l$  with zerotree  $Z$  and the Said-Pearlman encoder.

We next consider a collection  $Z_1, Z_2, \dots, Z_M$  of zerotrees. Using the quantization interpretation of a two-stage weighted universal code [3, 4], we consider this collection to be a *codebook* of zerotrees. Thus we define a "first-stage quantizer"  $\tilde{\beta} \circ \tilde{\alpha}$ , with encoder  $\tilde{\alpha}: \mathcal{X}^N \rightarrow \tilde{\mathcal{S}}$  and decoder  $\tilde{\beta}: \tilde{\mathcal{S}} \rightarrow \mathcal{C}$ . The first-stage quantizer maps the input space of possible data blocks  $x^N$  to the output space of possible zerotrees by way of the prefix code  $\tilde{\mathcal{S}}$ . We here assume that  $N$  is a multiple of  $l$ . The first-stage encoder chooses for each  $N$ -block a single zerotree. We then use the chosen zerotree to encode each of the  $l$ -vectors in  $x^N$ .

Using the above first-stage quantizer, the total distortion associated with encoding data block  $x^N$  with zerotree  $\tilde{\beta}(\tilde{\alpha}(x^N))$  is

$$d(x^N, \tilde{\beta}(\tilde{\alpha}(x^N))) = \sum_{i=1}^{N/l} d(x_i^l, \tilde{\beta}(\tilde{\alpha}(x^N))).$$

Likewise, the rate associated with encoding data block  $x^N$  with zerotree  $\tilde{\beta}(\tilde{\alpha}(x^N))$  is

$$r(x^N, \tilde{\beta}(\tilde{\alpha}(x^N))) = |\alpha(x^N)| + \sum_{i=1}^{N/l} r(x_i^l, \tilde{\alpha}(x^N)),$$

where the rate associated with the choice of a particular zerotree includes both the rate needed to describe the chosen zerotree and the rate used to describe the data given the zerotree described in the first stage.

Using a Lagrangian in order to minimize the distortion subject to a constraint on the rate, the optimal first-stage

encoder  $\tilde{\alpha}^*$  for a given  $\tilde{\beta}$  is

$$\tilde{\alpha}^*(x^N) = \arg \min_{\tilde{s} \in \tilde{\mathcal{S}}} [d(x^N, \tilde{\beta}(\tilde{s})) + \lambda r(x^N, \tilde{\beta}(\tilde{s}))]$$

for every  $x^N$ . Thus the optimal first-stage encoder chooses the code in its collection that achieves the best average Lagrangian performance in encoding the subvectors of a given vector  $x^N$ . We call the optimal first-stage encoder a *nearest neighbor* encoder.

The optimal first-stage decoder  $\tilde{\beta}^*$  for a given first-stage encoder  $\tilde{\alpha}$  satisfies

$$\tilde{\beta}^*(\tilde{s}) = \arg \min_Z E [d(x^N, \tilde{\beta}(\tilde{s})) + \lambda r(x^N, \tilde{\beta}(\tilde{s})) | \tilde{\alpha}(x^N) = \tilde{s}]$$

for every  $\tilde{s} \in \tilde{\mathcal{S}}$ . In this case, the optimal zerotree would be the zerotree that yields the lowest Lagrangian performance at slope  $\lambda$  on the distortion-rate curve. We call the process of designing the optimal first-stage decoder *decoding to the centroid*. Unfortunately, *optimal* zerotree design remains an open problem. We consider a new iterative technique for zerotree design in Section 3.

The code design algorithm employs an iterative descent technique to minimize the expected Lagrangian performance. We initialize the algorithm with an arbitrary prefix code  $\tilde{\mathcal{S}}$  and collection  $\{\tilde{\beta}(\tilde{s}) : \tilde{s} \in \tilde{\mathcal{S}}\}$  of zerotrees. Each iteration proceeds through three steps.

- 1 *Nearest Neighbor Encoding.*

Optimize the first-stage encoder  $\tilde{\alpha}$  for the given first-stage decoder  $\tilde{\beta}$  and prefix code  $\tilde{\mathcal{S}}$ .

- 2 *Decoding to the Centroid.*

Optimize the first-stage decoder  $\tilde{\beta}$  for the newly redesigned first-stage encoder and the given first-stage prefix code  $\tilde{\mathcal{S}}$ .

- 3 *Optimizing the Prefix Code.*

Optimize the first-stage prefix code  $\tilde{\mathcal{S}}$  for the newly redesigned first-stage encoder  $\tilde{\alpha}$  and decoder  $\tilde{\beta}$ . The optimal prefix code  $\tilde{\mathcal{S}}^*$  for a given first-stage encoder  $\tilde{\alpha}$  and decoder  $\tilde{\beta}$  is the entropy code matched to the probabilities  $P\{\tilde{\alpha}(X^N) = \tilde{s}\}$ , for which the ideal codelengths are  $|\tilde{s}^*| = -\log P\{\tilde{\alpha}(X^N) = \tilde{s}\}$ .

Each step of the algorithm decreases the (non-negative) expected Lagrangian performance, and thus the algorithm is guaranteed to converge.

While the system is designed for a target rate (controlled by changing  $\lambda$ ), it retains its embedded binary description and can therefore be used for early reconstructions and run past its target rate if desired by the end user. Thus the code is optimized for a target rate, but achieves good performance for a wide array of possible rates.

### 3 Zerotree Design

Given the wavelet (or wavelet packet) decomposition of a particular data set, we wish to order the coefficients

into the tree that best reflects the statistics of the given data. Tree design incorporates minimum threshold design, coefficient ordering, and tree branching rate. (The Said-Pearlman zerotree uses a quadtree structure and thus a branching rate of 4). Optimal zerotrees are expected to be data and rate dependent.

To design the optimal threshold, ordering, and branching rate, we must understand the tree properties that yield good coding performance. As discussed earlier, the success of zerotree codes relies primarily on two factors: the use of the shallowest possible tree for a given minimal threshold, and the use of a tree in which high energy coefficients deep in the tree are well-predicted by corresponding high energy coefficients closer to the tree's root. Notice that the need for the second factor kicks in when the first factor fails. That is, if all of the energy of the source is contained in the shallower levels of the associated zerotree, then the second factor becomes less significant. Recall also that the second factor alone does not yield good compression performance. That is, a tree that contains a lot of energy deep within its structure will not yield good performance even when that energy is perfectly predicted by lower levels of the tree. In designing our zerotrees, we will therefore concentrate on the first of these two factors. For simplicity, we do not include entropy coding in our zerotree codes.

The zerotree design algorithm is an iterative descent technique. We will initialize the zerotree with an arbitrary minimal threshold (or accuracy level) and branching rate. Notice that the expected distortion associated with a given zerotree is a function only of the minimal threshold employed in that zerotree. Thus the optimal ordering for a given threshold and branching rate is the ordering that minimizes the expected rate with respect to the data set. Roughly speaking, this expected rate is minimized by ordering the coefficients from largest to smallest. This ordering can be achieved in a number of different ways. Most simply, we can order the coefficients according to their average energy. Slightly better performance would be achieved by ordering the coefficients according to the probability with which that coefficient exceeds a given threshold.

Given a fixed ordering and minimal threshold, the optimal branching rate is the branching rate that achieves the lowest expected rate. For simplicity we assume a fixed branching rate within a given zerotree. Since the number of possible branching rates is relatively small, we here consider all possible branching rates and choose the branching rate that minimizes the expected rate with respect to a given minimal threshold and ordering.

Given a fixed ordering and branching rate, the optimal minimal threshold is the minimal threshold that results in

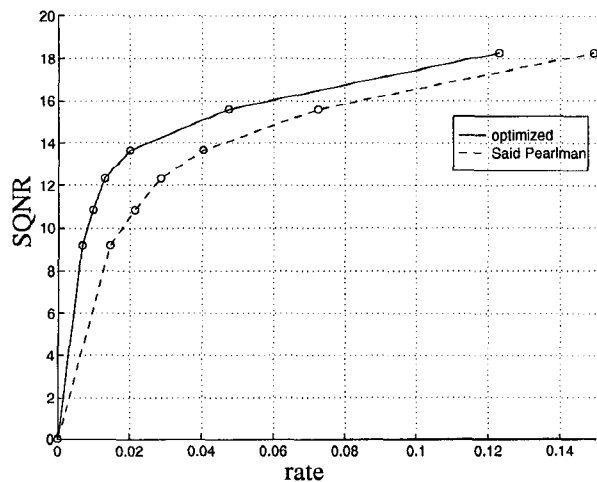


Figure 1: Comparison of SQNR results on a collection of combined text and gray scale images. The systems tested include a single optimized-zerotree code and the Said-Pearlman zerotree code.

the lowest expected Lagrangian performance for that tree. We here consider only thresholds of the form  $2^k$  for some integer  $k$  less than the logarithm of the maximal wavelet coefficient magnitude. As a result, the number of thresholds is likewise small, and the optimization of the threshold is performed by exhaustive search.

By iterating the above ordering, branching rate, and minimal threshold designs, we achieve a descent algorithm on the expected Lagrangian performance that can be iterated to convergence.

## 4 Experimental Results

In Figure 1, we compare the performance of a single optimized zerotree to the performance of a single Said-Pearlman zerotree. The optimized zerotree is designed using a single  $2400 \times 2400$  image scanned from a page of *IEEE Spectrum Magazine* and tested on another page from the same issue. Each page has roughly equal amounts of text and gray scale material. The optimized zerotree achieves up to 4 dB improvement over the Said-Pearlman zerotree. Incorporation of the zerotree design algorithm into the optimal weighted universal code design algorithm is therefore expected to yield even greater experimental performance gains.

## References

[1] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 6(3):243–250, December 1996.

[2] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

[3] P. A. Chou. Code clustering for weighted universal VQ and other applications. In *Proceedings of the IEEE International Symposium on Information Theory*, page 253, Budapest, Hungary, June 1991.

[4] P. A. Chou, M. Effros, and R. M. Gray. A vector quantization approach to universal noiseless coding and quantization. *IEEE Transactions on Information Theory*, IT-42(4):1109–1138, July 1996.

[5] M. Effros and P. A. Chou. Weighted universal bit allocation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 2343–2346, Detroit, MI, May 1995. IEEE.

[6] M. Effros and P. A. Chou. Weighted universal transform coding: universal image compression with the Karhunen-Loeve transform. In *Proceedings of the IEEE International Conference on Image Processing*, Washington, D.C., October 1995. IEEE.

[7] M. Effros and P. A. Chou. Universal image compression. 1996. Submitted to the *IEEE Transactions on Image Processing* December 18, 1996.